UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/746,978	12/21/2000	Pascal L. Renard	SC91211A	8364

7590 04/05/2004

EXAMINER
VO, TED T

Motorola, Inc
Austin Intellectual Property Law Section
7700 West Parmer Lane
MD: TX32/PL02
Austin, TX 78729

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 04/05/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

7

Office Action Summary	Application No. 09/746,978	Applicant(s) RENARD ET AL.
	Examiner Ted T. Vo	Art Unit 2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 January 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-11 and 16-28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11, 16-28* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the communication filed on 1/28/2004.

Claims 12-15 are canceled. Claims 20-28 are newly added.

Claims 1-11, 16-19 remain original claims. A new ground of rejection is provided to Claims 1-11. Accordingly, this office action is non-final.

Claims 1-11, 16-28 are pending in the application.

Response to arguments

2. Applicants' arguments to Claims 1-11 rejected under 35 U.S.C. 103(a) over Stoodley have been fully considered, but are moot in view of the new ground(s) of rejection.

Applicants' arguments to Claims 16-19 rejected under 35 U.S.C 101 have been fully considered. However, it is not persuasive.

For example, Applicants contend that Claiming to a processor instruction is not merely manipulation of an abstract idea, but has a practical application in an area of processor execution (Re: Remarks, page 7, paragraph "Rejection of Claim 14-19 under 35 U.S.C. 101").

Examiner responds: Claim 16-19 are labeling the data such as "fields" in an instruction. Such data does not impart functionality. It is nonfunctional descriptive material (see MPEP 2106 (IV)(B)(1)). Claims 16-19 do not produce any practical results, thus hold a data structure per se.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements of this title.

4. The claims 16-19 are rejected under 35 U.S.C 101 because the claimed invention is directed to non-statutory subject matter.

As per claims 16-19:

The claims 16-19 are rejected under 35 U.S.C 101 because the claimed invention is directed to non-statutory subject matter.

Claims 16-19 are merely a data structure per se that is not limited to practical applications.

Analysis:

Claim 16 is claiming, "*A processor instruction comprising: a first field that indicates a first termination condition for a first execution loop; and a second field that indicates a second termination condition for a second execution loop*". The claimed limitation is expressed as fields in a processor instruction. The fields without functionality to be realized are rather labels or merely "Nonfunctional descriptive material" than performing a practical application. Therefore, it holds the claim to be a data structure per se.

Claims 17-19, which further include more fields in the processor instruction, fail to remedy the deficiencies of independent claim 16.

According to the analysis above, claims 16-19 are claiming a software element "A processor instruction" that is not tangibly embodied for causing the computer to execute in a practical manner. The Claims are labeled with fields. The claims 16-19 thus are data structure per se and held nonstatutory.

To expedite a complete examination of the instant application the claims 16-19 rejected under 35 U.S.C. 101 (nonstatutory) above are further rejected as set forth below in anticipation of application amending these claims to place them within the four statutory categories of invention.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1, 3-5 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stoodley, "Software Pipelining Loops With Conditional Branches", 1996 IEEE, in view of Albert et al., "Data Parallel Computers and the FORALL Statement", 1990 IEEE.

As per claim 1:

Stoodley discloses, "***A processor comprising:***

an execution unit to execute a set of instructions (re: Stoodley: see page 269, figure 6); ***and***

an instruction fetching mechanism that retrieves the set of instructions to be executed by the execution unit (re: Stoodley: see page 269, figure 6, referring to the arrow 'From Code Memory'), ***at least one of the set of instructions*** (re: Stoodley: see pages 267-268, section 2.2, The Combine Algorithm, referring to VLIW instructions) ***comprising a single instruction*** (each VLIW is an single instruction) ***that provides for execution of other instructions of the set of instructions*** (re: Stoodley: see pages 265, left column, last paragraph) ***in accordance with multiple looping constructs***":

Stoodley does not explicitly disclose the execution of other instructions of the set of instructions "*in accordance with multiple looping constructs*". Its discussion of looping constructs is based on an example of a single loop in accordance with C code (re: Stoodley: Page 263; Figure 1). However, Stoodley teaches about single VLIW loop instruction that is likely executing from different iterations (re: Stoodley: page 265, left column, last paragraph), where the instruction VLIW combines many operations, some of operation perform iteration (re: Stoodley: see pages 265, left column, last paragraph; and see pages 267-268, section 2.2: The Combine Algorithm, and referring to VLIW instructions). These suggest about a single instruction provided for execution of other instructions of the set of instructions in accordance with multiple looping constructs.

Albert discloses multiple loop instruction "FORALL" (See Albert, particularly in page 391). This loop instruction results every processor when compiles and executes it would generate multiple loop instruction constructs.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to modify the operations of a single loop "VLIW instruction" provided with multiple iteration paths as in the teaching of Stoodley in accordance with the teaching "FOR ALL" instruction of Albert in a high level multiple loop instruction. Doing so, when converting a high level multiple loop instruction into a low level single instruction of VLIWs would result a single instruction VLIW provided for execution of other instructions of the set of instructions in accordance with multiple looping constructs; and thus that would conform to the set variables such as start/end or condition branches existed in such a high level multiple looping.

As per claim 3:

Stoodley does not disclose explicitly "*in accordance with multiple looping constructs*" as recited in claim 1. The combination of Stoodley and Albert is provided and the Claim 1 is rejection is provided.

Albert further discloses, "***wherein the multiple looping constructs are implemented in a nested structure***" (Re: Albert: page 391, referring to the loop instruction "FORALL"), where nested loop is defined commonly in computer programming.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include the nested structure as the "FORALL" instruction of Albert, compiled and executed into the VLIW instructions of Stoodley. Doing so would be implemented with a nested loop structure in low level programming constructs, and thus would correspond to the nested structure of FORALL.

As per claim 4: Regarding, "***The processor of claim 1, wherein the single instruction includes a plurality of loop termination conditions***",

Stoodley does not expressly shows ***plurality of loop termination conditions.***

However, Stoodley discloses VILW, a single instruction that combines the operations. Some of these operations perform multiple iterations and dealt with condition branches (re: Stoodley: see page 262, right column, the last full paragraph).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to correspond the condition branches in the VLIW in accordance to a given multiple loop instruction. Doing so would conform to the structural requirement of multiple looping.

As per claim 5: Regarding, "***The processor of claim 1, wherein the single instruction includes at least one field that identifies a location of a last instruction of a loop***", Stoodley teaches the identification inherently in branch directions using the flow diagram of figure 1.

7. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Stoodley, "Software Pipelining Loops With Conditional Branches", 1996 IEEE, in view of Albert et al., "Data Parallel Computers and the FORALL Statement", 1990 IEEE, and further in view of Tran (US 6,003,128).

As per claim 2: Claim 2 is further limitation of Claim 1.

The rejection of Claim 1 over Stoodley in view of Albert is provided above.

Stoodley and Albert do not expressly show further limitation of claim 2, *wherein the single instruction* ***initializes*** *a plurality of loops for later execution.*

Tran discloses such limitation (re: Tran: see column 1, lines 59-65: "The loop count is typical set by initializing a specified register prior to the execution of the loop").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to include initialization of loop execution in Tran to the single instruction VLIW that handles multiple iteration paths used by the processor of Figure 6 in Stoodley. Doing so would conform to the execution requirement of a processor that requires a value setting for use in its execution.

8. Claims 6-8 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stoodley, "Software Pipelining Loops With Conditional Branches", in view of Tran (US 6,003,128).

As per claim 6:

Stoodley discloses,

"***A method of performing an instruction for use by a processor, the method comprising:***

fetching the instruction from a memory source (re: Stoodley: See page 269, figure 6, "From Code Memory");

decoding the instruction to identify an instruction type (re: Stoodley: See page 269, figure 6, "Instruction Decoder"); *and*

initialization a plurality of dedicated loop storage elements corresponding to a plurality of different loops to be executed by

a single instruction (re: Stoodley: VLIW instruction).

Stoodley does not explicitly disclose ***initialization*** *a plurality of dedicated loop storage elements corresponding to a plurality of different loops to be executed by a single instruction*",

Tran discloses such a limitation (re: Tran: see column 1, lines 59-65: "The loop count is typical set by initializing a specified register prior to the execution of the loop").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to include initialization of loop execution in Tran to the single instruction VLIW that handles multiple iteration paths used by the processor of Figure 6 in Stoodley. Doing so would conform to the execution requirement of a processor that requires a value setting for use in its execution.

As per claim 7: Stoodley does not expressly show the storage elements include at least one of the following: "***an end-of-loop indicator, a start of loop indicator, a loop count, a loop register, and a condition code selection***".

Tran further discloses such limitation (re: Tran: see column 1, lines 59-65: "The loop count is typical set by initializing a specified register prior to the execution of the loop").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include at least one of loop variables in register for confirming to a structure of a given loop.

As per claim 8: Regarding, "**The method of claim 6, wherein the storage elements include a loop address for a start of a loop**":

Tran further discloses such a limitation (re: Tran: column 2, lines 1-10, "target address").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include loop address of Tran for confirming to a structure of a given loop.

9. Claims 9-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stoodley, "Software Pipelining Loops With Conditional Branches", 1996 IEEE.

As per claim 9:

Stoodley discloses, "***A method of performing an instruction for use by a processor, the comprising:***

fetching the instruction from a memory source (re: Stoodley: see figure 6, 'From Code Memory');

decoding the instruction to identify an instruction type (re: Stoodley: see figure 6, 'Instruction decoder'); *and*

determining a loop type for a single instruction that is to execute a plurality of different loops (see figure 6, 'Instruction decoder'),

the loop type comprising one of a conditional and nonconditional type of loop termination",

Stoodley does not explicitly address "*the loop type comprising one of a conditional and nonconditional type of loop termination*". However, Stoodley suggests conditional type (condition branch) and nonconditional type (iteration paths, such as C code loop in Figure 1) and including more one condition branch resulting in more than two iteration paths (re: Stoodley: see page 263, first column, second paragraph, 'If the execution...', particularly, see resulting in more than two iteration paths) and discusses difficulties of creating VLIW instructions containing operations from different iterations (re: Stoodley: see page 265, second column, third paragraph, 'The second difficulty...').

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to modify an instruction to perform multiple iteration paths with conditional or non conditional type in conforming to the logical requirements given in multiple loop instructions.

As per claim 10:

Stoodley discloses, "***A method of executing at least one instruction by a processor, the method comprising:***

fetching a first instruction from a memory source (re: Stoodley: see figure 6, 'From Code Memory');

decoding the first instruction to identify an instruction type (re: Stoodley: see figure 6, 'Instruction decoder'); ***and***

determining a loop type selected (re: Stoodley: see figure 6, 'Operations to Execute') ***from one of a conditional and nonconditional type of loop termination for a loop that contains more than instruction other than the first instruction***",

Stoodley does not explicitly address "*from one of a conditional and nonconditional type of loop termination for a loop*". However, Stoodley suggests conditional type (condition branch) and nonconditional type (iteration paths, such as C code loop in Figure 1) and including more one condition branch resulting in more than two iteration paths (re: Stoodley: see page 263, first column, second paragraph, 'If the execution...', particularly, see resulting in more than two iteration paths) and discusses difficulties of creating VLIW instructions containing operations from different iterations (re: Stoodley: see page 265, second column, third paragraph, 'The second difficulty...').

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to modify an instruction to perform multiple iteration paths with conditional or non conditional type in conforming to the logical requirements given in multiple loop instructions.

As per claim 11: Regarding, "***A method of executing at least one instruction by a processor, the method comprising:***

fetching a first instruction from a memory source (re: Stoodley: see figure 6, 'From code memory'); ***decoding the first instruction to identify an instruction type*** (re: Stoodley: see figure 6, 'Instruction decoder'); ***and***

determining a loop type selected (re: Stoodley: see figure 6, 'Operations to Execute') ***from one of a conditional and nonconditional type of loop termination for a loop that contains at least one instruction that may be interrupted during loop execution***",

Stoodley addresses a single looping construct that has condition branches with multiple iteration paths (re: Stoodley: see page 262, right column, the last full paragraph, referring to "a loop with B condition branches"), and show the processor to decode at least one instruction (see Figure 6).

Stoodley does not expressly show the decoder that determines the loop type "*from one of a conditional and nonconditional type of loop termination for a loop*". The suggestion of Stoodley with looping of more than one condition branch resulting in more than two iteration paths (re: Stoodley: see page 263, first column, second paragraph, 'If the execution...', particularly, see resulting in more than two iteration paths) and discusses difficulties of creating VLIW instructions containing operations from different iterations (re: Stoodley: see page 265, second column, third paragraph, 'The second difficulty...').

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to modify an instruction to perform multiple iteration paths with conditional or non conditional type in conforming to the logical requirements given in multiple loop instructions.

Furthermore, Stoodley does not explicitly disclose a loop that "*contains at least one instruction that may be interrupted during loop execution*". Stoodley shows a single instruction with a condition branch for performing at least two iteration paths (figure 1, block A).

Official notice is taken that an instruction that may be interrupted during loop execution is well known in the art. Interrupting is occurred by errors caused by the execution of that instruction or exception handling at branch instructions.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to include exception handling or to inject an error at the instruction. This would cause an interrupt during execution.

10. Claims 16-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Albert et al., "Data Parallel Computers and the FORALL Statement", 1990 IEEE.

As per claim 16:

Albert discloses a data structure, "FORALL (.,. ,.,.) (re: Albert: see page 391, left column, data structure 'FORALL') to cover the teaching, "***A processor instruction comprising: a first field that indicates a first termination condition for a first execution loop*** (re: Albert: see page 391, left column, referring instruction, "FORALL" with field 'I=1:N'); ***and a second field that indicates a second termination condition for a second execution loop*** (re: Albert: see page 391, left column, referring instruction "FORALL" with field 'J=1:M').

Albert does not address the data structure in a manner of a processor instruction. However, it would implement the Albert's instruction for processor instruction because of the similarity of loop description.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to intend use a similar structure of Albert for a processor instruction. Doing so would conform to a looping structure.

As per claim 17: Albert discloses "***The processor instruction of claim 16, wherein the first execution loop is a same loop as the second execution loop***" (re: Albert: see page 391, left column, particularly, "FORALL" with I and J).

As per claim 18: Albert discloses "***The processor instruction of claim 16, further comprising a third field that indicates a first end-of-loop location***" (re: Albert: see page 391, left column, particularly, "FORALL" with 'N').

As per claim 19: Albert discloses "***The processor instruction of claim 18, further comprising a fourth field that indicates a second end-of-loop location***" (re: Albert: see page 391, left column, particularly, "FORALL" with 'M').

11. Claims 20-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stoodley, "Software Pipelining Loops With Conditional Branches", in further view of Albert et al., "Data Parallel Computers and the FORALL Statement", 1990 IEEE and Tran (US 6,003,128).

As per claim 20:

Stoodley discloses, "***A method of performing an instruction for use by a processor, the method comprising:***

fetching a single instruction from a memory source (re: Stoodley: See page 269, figure 6, referring to the arrow 'From Code Memory'), ***wherein the single instruction provided for execution of multiple looping constructs, the multiple looping constructs including a first loop and a second loop to be executed;***

decoding the single instruction (re: Stoodley: see figure 6, 'Instruction decoder'); ***and***

in response to decoding the single instruction, using information provided by the single instruction to initialize a plurality of loop storage elements corresponding to the first loop and the second loop":

Stoodley discloses an execution mechanism (re: Stoodley: see, figure 6) for fetching instruction from memory source. The mechanism includes the fetch of a single instruction VLIW (re: Stoodley: Figure 6, "From Code Memory" [fetching]) that is dealt with branches (Re: Stoodley: pages 267-268, section 2.2) or loop iteration paths. In response to decoding the code fetched from the memory, the decoder sets decoded information in the instruction to registers (re: Stoodley: see Figure 6, particularly the registers that are coupled with AU0, IU0, and FU0).

- Stoodley does not expressly show the single instruction VILW *provided for execution of multiple looping constructs, the multiple looping constructs including a first loop and a second loop to be executed.*

However, Stoodley teaches about single VLIW instruction that will most likely be executing operations from different iterations (re: Stoodley: page 265, left column, last paragraph), where the single instruction VLIW are combined by many operations (See pages 265, left column, last paragraph; and see pages 267-268, section 2.2, The Combine Algorithm, referring to VLIW instructions). Some of these operations perform branching. These suggest about a single instruction provided for execution of other instructions of the set of instructions in accordance with multiple looping constructs; and

- Stoodley does not expressly address *using information provided by the single instruction to initialize a plurality of loop storage elements corresponding to the first loop and the second loop.*

Albert discloses multiple loop instruction "FORALL" (See Albert, particularly in page 391). This loop instruction results every processor when compiles and executes it would generate multiple loop instruction constructs; and

Tran discloses the limitation *using information provided by the loop instruction to initialize a plurality of loop storage elements* (re: Tran: see column 1, lines 59-65: "The loop count is typical set by initializing a specified register prior to the execution of the loop").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to include a multiple loop of Albert at high level programming into the execution mechanism as disclosed by Stoodley and the teaching about loop initialization of Tran.

Doing so, when converting a high level multiple loop instruction (for example, FORALL) into a low level single instruction of VLIWs, would result a single instruction VLIW provided for execution of other instructions of the set of instructions in accordance with multiple looping constructs; and thus that would conform to the execution requirement of a processor that requires a value setting for use in its execution.

As per claim 21: Regarding "***The method of claim 20, wherein the multiple looping constructs are nested such that the first loop is nested within the second loop***":

Stoodley does not explicitly disclose "*in accordance with multiple looping constructs*" as recited in claim 20. The combination of Stoodley and Albert is provided and the Claim 1 is rejection is provided.

Albert further discloses, "*wherein the multiple looping constructs are implemented in a nested structure*" (Re: Albert: see loop structure "FOR ALL", page 391), where nested loop is defined commonly in computer programming.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include the nested structure such as "FOR ALL" instruction of Albert into the processor execution of Stoodley. Doing so would be implemented with a nested loop structure in low level programming constructs for confirming to nested looping.

As per claim 22: Regarding "***The method of claim 20, wherein the information provided by the single instruction to initialize the plurality of loop storage elements includes at least one termination condition for each of the first loop and the second loop.***

Trans further discloses the limitation (re: Tran: see column 1, lines 59-65: "The loop count is typical set by initializing a specified register prior to the execution of the loop").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include loop terminations for conforming to the loop structure requirements.

As per claim 23: Regarding "***The method of claim 22, wherein at least one termination condition for each of the first loop and the second loop comprises at least one of a loop count value and a condition code"***

Trans further discloses the limitation (re: Tran: see column 1, lines 59-65: "The loop count is typical set by initializing a specified register prior to the execution of the loop").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include loop count value and condition code for conforming to the loop structure requirements.

As per claim 24: Regarding "***The method of claim 22 further comprising:***

performing a first set of logic relating to the at least one termination condition for the first loop; and performing a second set of logic relating to the at least one termination condition for the first loop

Trans further discloses the limitation (re: Tran: column 3, lines 25-46, also see column 30, lines 60-67 and column 31, lines 1-11, "Nested loop structure").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include a performance of each loop count value and condition code for conforming to the nested loop structure requirements.

As per claim 25: Regarding to limitation of claim 25, the claim recitation describes a common multiple loop structure, and is included with "***an end of loop indication for each of the first loop and the second loop***" that has the similar functionality as recited in claim 22: "***at least one terminal condition for each of the first loop and the second loop***".

Thus, Claim 25 is rejected in the same reason as set forth in connecting to the rejection of Claim 22.

As per claim 26: Regarding to limitation of claim 26, claim recitation describes a common multiple loop structure, Tran further discloses the limitation (re: Tran: see column 30, lines 60-67 and column 31, lines 1-11, "Nested loop structure").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include further description of nested loop in one single instruction of Stoodley for conforming to the nested loop structure definition.

As per claim 27: Regarding to limitation of claim 26, claim recitation describes a common multiple loop structure, Tran further discloses the limitation (re: Tran: see column 30, lines 60-67 and column 31, lines 1-11, "Nested loop structure").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further include further description of nested loop in one single instruction of Stoodley for conforming to the nested loop structure definition.

As per claim 28: Stoodley does not expressly show "**determining a loop type corresponding to the single instruction, wherein *the loop type is selected from one of a conditional and nonconditional type***".

However, Stoodley suggests conditional type (condition branch) and nonconditional type (iteration paths, such as C code loop in Figure 1) and including more than one condition branch resulting in more

than two iteration paths (re: Stoodley: see page 263, first column, second paragraph, 'If the execution...', particularly, see resulting in more than two iteration paths) and discusses difficulties of creating VLIW instructions containing operations from different iterations (re: Stoodley: see page 265, second column, third paragraph, 'The second difficulty...').

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to further modify an instruction to perform multiple iteration paths with conditional or non conditional type in conforming to the conditions of a given multiple loop structure.

Conclusion

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (703) 308-9049. The examiner can normally be reached on Monday-Friday from 8:00 AM to 5:30 PM ET. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam, can be reached on (703) 305-4552.
The fax phone numbers:

(703) 872-9306 (for formal communication intended for entry);

(703) 746-5429 (for informal or draft communication, please label "PROPOSED" or "DRAFT").

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Group receptionist whose telephone number is (703) 305-3900.

TED T. VO

Patent Examiner
Art Unit: 2122
March 31, 2004